УДК 510.5; 004.93

А. И. Абрамов, кандидат технических наук, ИжГТУ имени М. Т. Калашникова
И. В. Абрамов, доктор технических наук, профессор, ИжГТУ имени М. Т. Калашникова
Т. А. Мазитов, аспирант, ИжГТУ имени М. Т. Калашникова
А. М. Пальмов, аспирант, ИжГТУ имени М. Т. Калашникова

ПРИМЕНЕНИЕ ПЧЕЛИНОГО АЛГОРИТМА ДЛЯ ОБРАБОТКИ ДАННЫХ ЛАЗЕРНОЙ СКАНИРУЮЩЕЙ СИСТЕМЫ ПРИ НАВИГАЦИИ МОБИЛЬНЫХ РОБОТОВ

Введение

В мобильной робототехнике, при управлении устройствами, задача навигации и составления карты в неизвестной окружающей среде – SLAM(от англ. *Simultaneous Localization Mapping*) является одной из главных. Первые алгоритмы для решения данной задачи были предложены Леонардом и Дюран-Уайатом [1]. Их алгоритмы служили для одновременного оценивания положения вновь воспринимаемых ориентиров и положения самого мобильного робота во время построения карты. Многие исследователи успешно расширили SLAM в помещении [2], на открытом воздухе [3], в подземных [4, 5] и подводных [6] средах, в двумерном и трехмерном пространствах [7, 8].

Формирование карты в SLAM-алгоритмах основано на сравнении данных с сенсоров объекта (данные одометрии, данные лидара или сонара, показания гироскопа, RGB-D и телекамер и др.) в дискретные моменты времени: с некоторой точностью вычисляется перемещение объекта, далее карта дополняется новыми данными. Сопоставление данных характеризуется большой вычислительной сложностью. Точность построения карты в большой степени зависит от точности вычисления положения объекта. Существующие алгоритмы сопоставления данных можно разделить на два типа в зависимости от используемых ими входных данных: использующие различные особенности (линии, особые точки) и использующие «сырые» данные.

Одним из главных этапов при построении карты является нахождение траектории движения объекта по данным сенсоров. Он характеризуется большой вычислительной сложностью и занимает значительную часть всего алгоритма. Среди перечисленных сенсоров все больший интерес вызывают лидары, что обусловлено достоинствами, такими как высокая скорость и точность полученных данных. Данное исследование направлено на увеличение эффективности и скорости совмещения данных при использовании только данных лидара.

Математическая модель

Положение объекта на плоскости в каждый момент времени задается матрицей R:

$$R(k)=\begin{bmatrix} x_0(k) \\ y_0(k) \\ \varphi_0(k) \end{bmatrix}, \qquad (1)$$

где $x_0(k)$, $y_0(k)$ – координаты лидара в текущий момент в глобальной системе координат; $\varphi_0(k)$ – ориентация лидара в текущий момент в глобальной системе координат.

Для получения информации используется датчик HOKUYO utm-30lx-ew, предоставляющий данные дальности с частотой 40 Гц в диапазоне –135°…135° с шагом 0,25°, которые можно представить следующим вектором:

$$Z_l(k)=\{x_1(k), x_2(k), \ldots, x_{1081}(k)\}, \qquad (2)$$

где $Z_l(k)$ при l = 1…1081 – точка, заданная в полярных координатах в момент времени k.

Графически пример получаемых данных представлен на рис. 1.

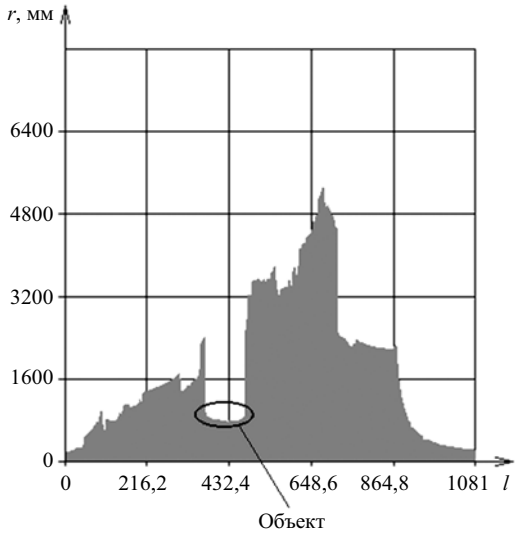


Рис. 1. Гистограмма, отображающая данные лидара

На рис. 1 в относительно плавно изменяющихся данных, описывающих периметр комнаты, четко выделяется провал. Этот массив точек характеризует объект, находящийся в наблюдаемом пространстве.

Представление точек в декартовой системе координат отражается в следующем виде:

$$Z^*_{\text{локальное}}(k)=\begin{bmatrix} x_l(k) \\ x_l(k) \end{bmatrix}=\begin{bmatrix} x_l(k)\cdot\cos\varphi_l \\ x_l(k)\cdot\sin\varphi_l \end{bmatrix} \qquad (3)$$

при l = 1...1081.

На рис. 2 представлены данные датчика в декартовой системе координат.

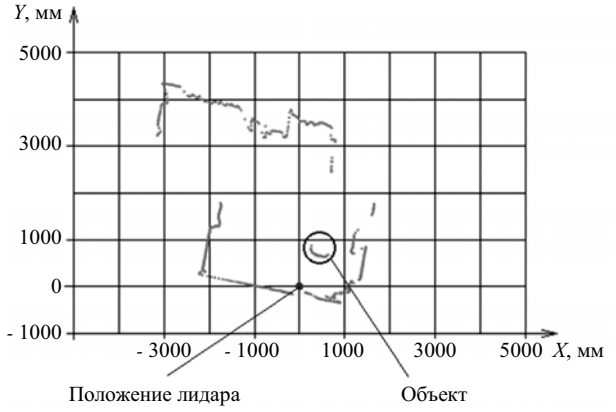


Рис. 2. Данные лидара в декартовой системе координат

Так как лидар предоставляет данные относительно своего положения, то он находится в центре локальной системы координат.

Данные датчика в глобальной системе координат будут определяться как

$$Z^{*}_{\text{глоб}}(k) = \begin{bmatrix} x_l(k)\cdot\cos\varphi_0 - y_l(k)\cdot\sin\varphi_0(k) + x_0(k) \\ x_l(k)\cdot\sin\varphi_0 - y_l(k)\cdot\cos\varphi_0(k) + y_0(k) \end{bmatrix} = \begin{bmatrix} x_{l\text{глоб}}(k) \\ x_{l\text{глоб}}(k) \end{bmatrix} \quad (4)$$

при $l = 1\ldots1081$.

Таким образом, задача сводится к нахождению угла поворота, величины и направления смещения облака точек, которые описывают движение объекта, по данным лидара. Для этого необходимо совместить облако точек, полученное в текущий момент времени, с предыдущим набором данных.

Пчелиный алгоритм

Основным инструментом для сопоставления облаков точек (нахождения матриц вращения и перемещения) является алгоритм, разработанный Y. Chen, G. Medioni (1992) [9], P.J. Besl, N.D. McKay (1992) [10] – итеративный поиск ближайшей точки – ICP (от англ. *Iterative Closest Point*). Пары ближайших точек исследуемых облаков связываются и находится преобразование, минимизирующее расстояние между парами. Операция повторяется многократно, пока не будет найдено приемлемое решение.

Одной из основных особенностей ICP-алгоритма является ограничение области сходимости: работает только при условии, что облака точек не сильно сдвинуты относительно друг друга. В противном случае алгоритм может найти локальный минимум. Таким образом, для обеспечения наибольшей точности в реальном времени необходимо обрабатывать все данные, так как различия между двумя последующими сканами минимальны. Также для алгоритма характерна большая вычислительная сложность: сопоставление пар и поиск смещения, минимизирующего суммарную среднеквадратическую ошибку между ближайшими соответствующими точками, занимает значительное время. Поэтому для минимизации функции различия было предложено использовать пчелиный алгоритм.

Пчелиный алгоритм – один из алгоритмов, описывающих коллективное поведение децентрализованной самоорганизующейся системы. Впервые термин был введен Beni G. и Wang J. в 1989 г. [11].

На первом шаге алгоритма в точки, описываемые случайными параметрами, отправляется некоторое количество пчел-разведчиков *S*. В зависимости от значения целевой функции, которое определяется координатами пчелы, выделяются перспективные участки на поверхности функции вблизи которых возможно располагается глобальный максимум.

- Выбирается *n* лучших участков, где значения целевой функции больше всех.
- Выбирается *m* так называемых выбранных участков, где значения целевой функции меньше, чем на «лучших» участках, но эти участки также являются «неплохими» с точки зрения значения целевой функции.

В окрестность *n* «лучших» участков посылается *N* пчел, а в окрестность *m* выбранных участков посылается *M* пчел.

На каждом следующем шаге алгоритма уменьшается окрестность участков, в которые высылаются пчелы, пока не будет найдено значение функции с заданной точностью.

Алгоритм в несколько раз увеличивает количество вычислений по отношению со стандартными методами минимизации (алгоритм градиентного спуска и др.), но вычисления функции в случайных точках являются аналогичными операциями, что дает возможность эффективного использования параллельных вычислений, что в конечном счете приводит к увеличению быстродействия.

Итеративная модификация пчелиного алгоритма для совмещения облаков точек

Применение роевого пчелиного алгоритма позволило создать следующий алгоритм.

1. При поступлении новых данных происходит инициализация 256 пчел (обусловлено возможностями используемой видеокарты). Каждая из них обладает тремя параметрами: *x*, *y*, описывающие смещение вдоль осей, и α – угловое смещение датчика, причем для первой пчелы все параметры равны 0, а для остальных выбираются случайные значения в диапазоне, зависящем от максимальной скорости робота.

2. Параллельно вычисляется функция $R(x, y, \alpha) = \Sigma_i d_i^2$ для каждой пчелы: решается задача поиска ближайшего соседа.

3. Выбирается одна пчела с наименьшим значением – $\min R(x, y, \alpha)$.

4. Если значение функции больше заданного порога (не достигнута определенная точность), то в окрестность этой точки снова «высылаются» новые пчелы. Переход на шаг 2.

5. После нахождения минимума функции R на карту добавляются новые точки в соответствии с вычисленными параметрами сдвига и поворота.

6. Переход на шаг 1.

В графическом виде алгоритм представлен на рис. 3.

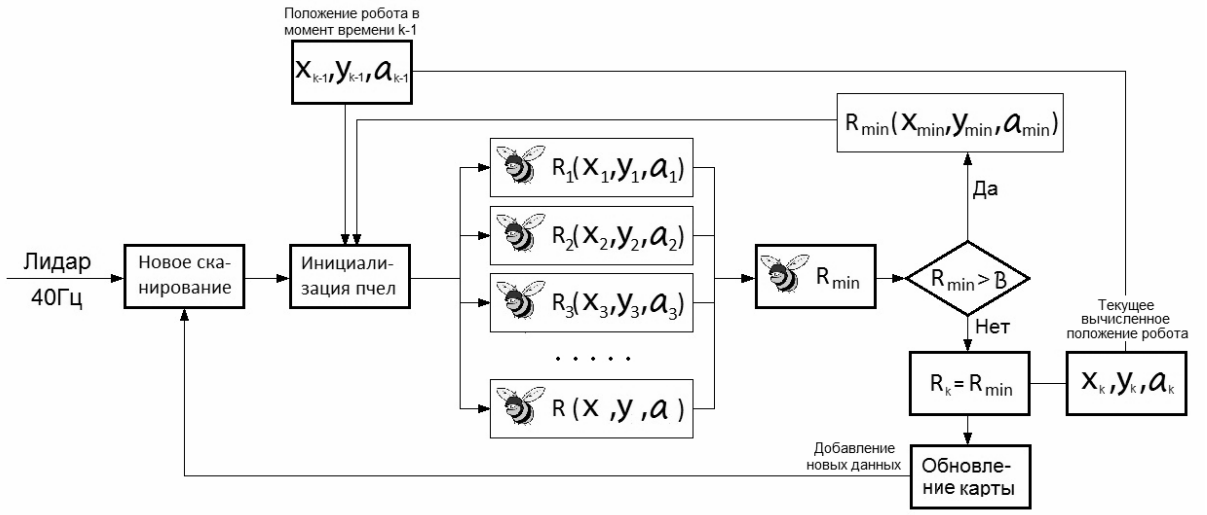


Рис. 3. Схема разработанного алгоритма

Параллельные вычисления, выполняемые на шаге 2, наглядно представлены на схеме алгоритма (рис. 3) – изображены в виде пчел.

В разработанном виде итеративный пчелиный алгоритм позволяет эффективно решать задачу совмещения облаков точек в двумерном пространстве. Аналогично может быть решена задача и для трехмерного пространства.

Результаты экспериментов

Эксперименты проводились в лаборатории технического зрения Ижевского государственного технического университета имени М. Т. Калашникова кафедры «Мехатроника и робототехника». Лидар был установлен на трехколесный мобильный робот (рис. 4), по каналу WiFi данные передавались на ПК, где производились все вычисления.

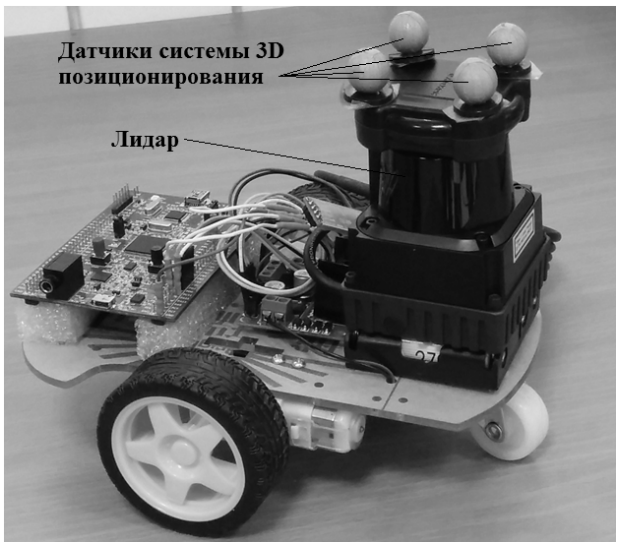


Рис. 4. Трехколесный мобильный робот, оснащенный лидаром

Робот перемещался по лаборатории и составлял карту по разработанному методу. На рис. 5 представлена полученная карта. Ее разрешение составляет 4 сантиметра на пиксел. Размер карты 600×600 пикселов.

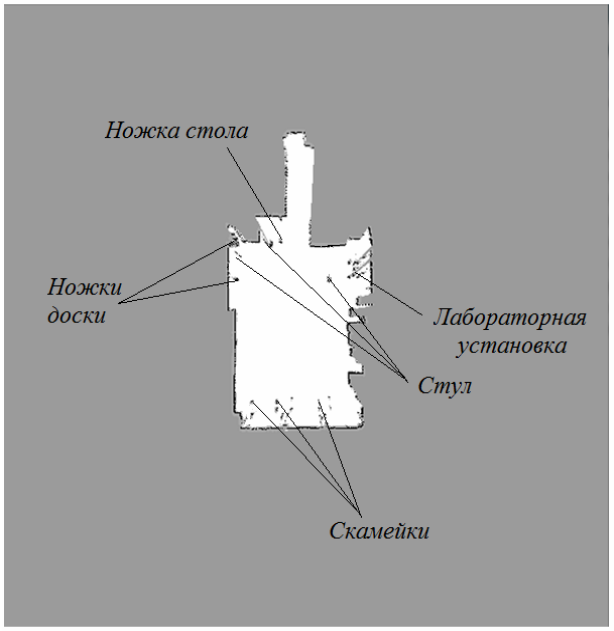
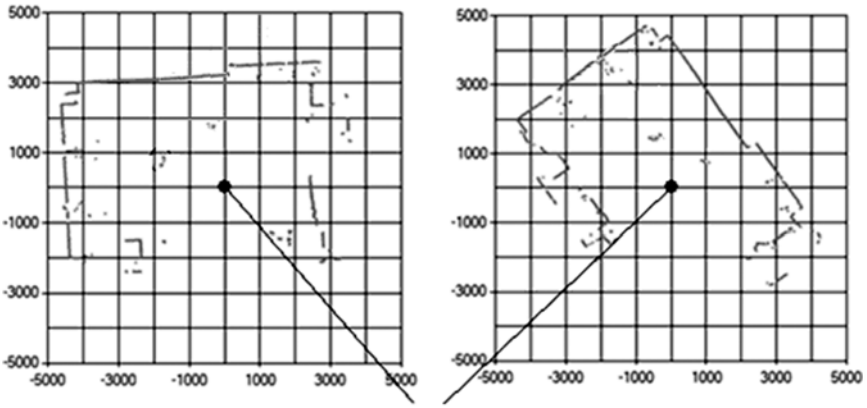


Рис. 5. Полученная карта лаборатории

Для оценки точности карты и позиционирования робота была использована система Vicon, позволяющая с точностью до 1 мм определить положение исследуемого объекта в 3D-пространстве. В эксперименте сравнивалось перемещение робота, измеренное системой позиционирования, и перемещение по составленной карте. На рис. 6 представлены данные лидара в тестовых позициях 1 и 2.



а *б*

Рис. 6. Данные лидара: *а* – в положении 1, *б* – в положении 2

При перемещении робота из точки 1 в точку 2 согласно построенной карте устройство переместилось по оси X – 320 мм, по оси Y – 720 мм; согласно системе позиционирования Vicon перемещение по оси X – 329 мм, по оси Y – 728 мм. Положение робота вычисляется путем совмещения карты с данными лидара, и поскольку построенная карта имеет разрешение 40 мм на пиксел, то и погрешность положения исследуемого объекта составляет 40 мм.

Средняя скорость совмещения облаков двумерных данных составила 3 итерации.

Таким образом, эксперимент подтвердил целесообразность использования итеративного пчелиного алгоритма для обработки данных лидара о положении мобильного робота в искомый момент времени. Для повышения точности оценки положения объекта при использовании данного алгоритма возможно увеличение разрешения карты, применение фильтрации данных лидара и т. д. Слабым местом алгоритма, как и для ICP, является вычисление функций различия между сканами – поиск ближайшей точки занимает значительное время.

Заключение

Применение итеративного пчелиного алгоритма, адаптированного для параллельных вычислений при совмещении облаков точек, позволило существенно увеличить скорость обработки данных. Среднее число итераций для нахождения угла поворота и смещения между двумя последующими сканами составило 3 итерации, в то время как для стандартных методик совмещения облаков точек [13] требуется от 10 до 30.

В случае навигации мобильного объекта по заранее известной карте коррекция положения будет состоять только из операции чтения значения соответствующей ячейки карты, и метод совмещения облаков точек, предложенный в статье, будет обеспечивать наибольшее быстродействие.

Библиографические ссылки

1. *Leonard J. J.*, *Durrant-Whyte H. F.* Simultaneous map building and localization for an autonomous mobile robot // Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on. – IEEE, 1991. – С. 1442–1447.

2. *Gallegos G.*, *Rives P.* Indoor SLAM based on composite sensor mixing laser scans and omnidirectional images // Robotics and Automation (ICRA), 2010 IEEE International Conference on. – IEEE, 2010. – С. 3519–3524.

3. SLAM in a dynamic large outdoor environment using a laser scanner / Zhao H. [at al.] // Robotics and Automation, 2008. ICRA 2008. IEEE International Conferenceon. – IEEE, 2008. – С. 1455–1462.

4. *Marshall J. A.*, *Barfoot T. D.* Design and field testing of an autonomous underground tramming system //Field and Service Robotics. – Springer Berlin Heidelberg, 2008. – С. 521–530.

5. *Marshall J.*, *Barfoot T.*, *Larsson J.* Autonomous underground tramming for center-articulated vehicles // Journal of Field Robotics. – 2008. – Т. 25, № 6-7. – С. 400–421.

6. *Ribas D. et al.* Underwater SLAM in man-made structured environments // Journal of Field Robotics. – 2008. – Т. 25, № 11-12. – С. 898–921.

7. A flexible and scalable slam system with full 3d motion estimation / Kohlbrecher S. [et al.] // Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium. – IEEE, 2011. – С. 155–160.

8. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems / Wurm K. M. [et al.] // Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation. – 2010. – Т. 2.

9. *Chen Y.*, *Medioni G.* Object modelling by registration of multiple range images //Image and vision computing. – 1992. – Т. 10, № 3. – С. 145–155.

10. *Besl P. J.*, *McKay N. D.* Method for registration of 3-D shapes // Robotics-DL tentative. – International Society for Optics and Photonics, 1992. – С. 586–606.

11. *Beni G.*, *Wang J.* Swarm intelligence in cellular robotic systems //Robots and Biological Systems: Towards a New Bionics. – Springer Berlin Heidelberg, 1993. – С. 703–712.

12. *Karaboga D.*, *Akay B.* A comparative study of artificial bee colony algorithm // Applied Mathematics and Computation. – 2009. – Т. 214, № 1. – С. 108–132.

13. *Rusinkiewicz S.*, *Levoy M.* Efficient variants of the ICP algorithm // 3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on. – IEEE, 2001. – С. 145–152.

Получено 20.04.2016